FPGAで組み込みOSを活用する

―― Nios IIとμClinuxを実装したシステムLSIの実現

中根隆康，福島雅史

ここでは，FPGAに実装したソフト・マクロのCPUを使ってLinuxを動作させる．CPUコアとして米国Altera社の「Nios II」，Linuxディストリビューションとしては「μClinux」を利用する．FPGAボードでLinuxカーネルを動作させた後，CompactFlashから起動できるようにする．（編集部）

最近，組み込みシステムの世界でもLinuxを活用する場面が増えています．Linuxの良いところは，オープン・ソースで誰でも自由に使えることと，コミュニティで作られているドライバなどを利用できることです．しかし，コーディング・ルールや試験方法などが明確に定められていないため，これらのソース・コードは，一貫性に欠けているように見えます．また，Linuxをビルドするためのmakefileも構造が複雑になり過ぎていて，初めてLinuxを導入しようとする方は少々戸惑うかもしれません（同じオブジェクトを何度もコンパイルするような構造のものもある）．

それでもLinuxが備えているネットワーク系のプロトコル・スタックやサーバ・アプリケーションを利用できるのは，かなりの魅力といえます．

1．ソフト・マクロのCPUと組み込みLinux

今回は，米国Altera社のソフト・マクロのCPU「Nios II」で，組み込みLinuxを動作させます注1．

Nios IIは32ビットの命令セット，32ビットのデータ・バス，32個の外部割り込みソース，最大2Gバイトの外部アドレス空間，最大256個のカスタム命令を持ったRISCプロセッサです．高速版（Nios II/f），標準版（Nios II/s），エコノミ版（Nios II/e）の3種類の構成を選択できます．今回は，Nios II/sを使いました．

ソフトウェア開発は，EclipseとGNUのコンパイラをベースとした総合開発環境（IDE）で行います．

組み込みLinuxには，いろいろなディストリビューションがあります．しかしNios II用となると限られてしまいます．今回は一番実績がありそうなμClinuxを搭載します．

● MMUを持たないマイコン向けのLinux

μClinuxは，Embedded Linux / MicrocontrollerProject（http://www.uclinux.org/）によって，米国Motorola社（現在は米国Freescale Semiconductor社）のDragon Ball（MC68328）を搭載するPDA「PalmPilot」向けに開発されたものです．

Linuxは仮想メモリを使うことが前提になっているため，MMU（Memory Management Unit）を持たないプロセッサでは動作させられませんでした．この仮想メモリを物理ページ管理に変更して，MMUを持たないプロセッサでも動作できるようにしたものがμClinuxです．そのため，残念ながらfork（プロセスのクローン生成）というUNIX系ではよく使われる機能が削除されています．

μClinuxの構造を**図1**に示します．

注1：米国Xilinx社のMicroBlazeを使った組み込みLinuxシステムには，例えば，アットマークテクノの「SUZAKU（朱雀）」がある．

KeyWord　FPGA，ソフト・マクロ，CPUコア，Nios II，μClinux，カーネル，CompactFlash

5

● Nios II対応のμClinux

Nios IIに対応するμ Clinuxは2種類あります．

- uclinux.org（http://www.uclinux.org/）からダウンロードできるフル・ソース・ディストリビューション（μClinux-dist-20070130.tar.gz）
- Nios ForumのμClinux Forumからダウンロードできる Nios II IDEのプラグイン（nios2linux-1.4.zip）

Nios Forumのほうは，本稿執筆時にはNios II IDEバージョン5.0をターゲットにしたものです注2．このため，最新のバージョン7.1では利用できません．そこで今回は，uclinx.orgのものを利用します（p.58のコラム「μClinuxの入手と開発環境のセットアップ」を参照）．

uclinx.orgのμClinuxを実装する際には，開発環境に注意しなければなりません．ハードウェアの設計はWindowsで動作するQuartus IIを使えますが，ソフトウェアはLinux環境を使う必要があります（p.59のコラム「開発環境について」を参照）．Windows環境しかない場合は，coLinuxが必要です(2)．

Windowsのファイル・システムがファイル名の大文字，小文字を区別しないためです．µclix.orgのμClinuxのディストリビューションの中には，同じ名前で大文字と小文字のファイルが存在しています．

<table>
<tr><td colspan="5">アプリケーション</td></tr>
<tr><td colspan="5">システム・コール・インターフェース</td></tr>
<tr><td>ターミナル</td><td>ファイル・システム</td><td>ネットワーク</td><td>IPC</td><td>プロセス管理</td></tr>
<tr><td>仮想メモリ管理</td><td colspan="2">ページ管理</td><td colspan="2">スケジューラ</td></tr>
<tr><td colspan="5">ハードウェア依存部</td></tr>
<tr><td colspan="5">ハードウェア（MMUなし）</td></tr>
</table>

図1　μClinuxシステムの構造
灰色の部分がμ Clinuxカーネル．

注2　Microtronix Datacom社のホームページ（http://www.microtronix.com/）にバージョン7.0版が一度リリースされたが，すでになくなっている．理由は不明．

コラム　μ Clinuxの入手と開発環境のセットアップ

● GNU開発環境

μClinuxのビルドにはGNUの開発環境を使用します．通常ではNios II用にクロス開発環境を構築する必要がありますが，Nios Community Wikiにバイナリが用意されています（http://nioswiki.jot.com/WikiHome/OperatingSystems/BinaryToolchanのnios2gcc.tar.bz2）．

● μClinuxディストリビューション

今回使用するディストリビューションはuclinux.orgで公開されているものです（http://www.uclinux.org/pub/uClinux/dist/のuClinux_dist20070130.tar.gz）．

さらにNios Community WikiにあるNios II用のパッチ・ファイルを適用します
（http://nioswiki.jot.com/WikiHome/OperatingSystems/UClinuxDistのuClinux-dist-20070130-nios2-02.diff.gz）．

```
ファイル(F)  編集(E)  表示(V)  端末(T)  タブ(B)  ヘルプ(H)

[fukushima@linux ~]$ nios2-linux-uclibc-gcc -v
Reading specs from /opt/nios2/lib/gcc/nios2-linux-uclibc/3.4.6/specs
Configured with: /root/buildroot/toolchain_build_nios2/gcc-3.4.6/configure --pre
fix=/opt/nios2 --build=i386-pc-linux-gnu --host=i386-pc-linux-gnu --target=nios2
-linux-uclibc --enable-languages=c --enable-shared --disable-__cxa_atexit --enab
le-target-optspace --with-gnu-ld --disable-nls --enable-threads --disable-multil
ib --enable-cxx-flags=-static
Thread model: posix
gcc version 3.4.6
[fukushima@linux ~]$ ▌
```

図A　nios2-linux-uclibc-gcc バージョン情報

● GNUツールのセットアップ

次にダウンロードしたツールをインストールします．ツールを保存したディレクトリへ移動し，管理者権限でログインし，以下のコマンドを実行します（#はコンソールのプロンプト）．

```
# tar jxf nios2gcc.tar. bz2 -C/
```

ツールは/opt/nios2以下に展開されます．

次に環境変数の設定を行います．.bash_profileに，

```
export PATH＝$PATH:/opt/nios2/bin
```

を追加します．また，よく使用するsdk_shell（Altera社が提供する開発ツールに付属するNois IIのコマンド・シェル）も追加します．

```
export PATH＝$PATH:/opt/Altera71/NiosIIEDS
```

環境変数の設定を適用させるためには，Linuxパソコンの再起動が必要です．確認のため，

```
# nios2-linux-uclibc-gcc -v
```

を実行します．**図A**のようなメッセージが表示されれば，インストールは完了です．

● ディストリビューションの解凍

カーネルの解凍とビルドは一般ユーザ権限で実行します．

まず，μClinuxのディストリビューションを解凍するディレクトリを作成します．筆者は/home/nios2_uClinuxディレクトリを用意しました．μClinuxディストリビューション・ファイルをnios2_uClinuxへ移動し，解凍します．

```
# tar zxvf uClinux-dist20070130.tar.gz
```

/home/nios2_uClinuxの下にuClinux-distディレクトリが作成され，その下にディストリビューション一式が解凍されます．

2. μClinuxをFPGA単体で動かしてみる

Nios IIでμClinuxを動かすまでの手順の概要を図2に示します．

ハードウェアとして，今回は，Altera社の「Nios II開発キットStratixエディション」に付属するFPGAボードを使用します．FPGAに実装する回路（Nios IIを含む）としては，Nios II EDS（EDS：Embedded Design Suite）に付属するサンプル・プロジェクトstandardを使います．.sofファイルを以下の手順でフラッシュ・メモリに書き込んでおきます．

```
$sof2flash--input=NiosII_stratix_
1s10_standard.sof--output=NiosII_
stratix_1s10_standard.flash--
offset0x600000
```

```
$nios2-flash-programmer--base=
0x00000000 NiosII_stratix_1s10_
standard.flash
```

● カーネルの設定

作業ディレクトリ（今回はμClinux-distとする）に移動し，Nios II用のパッチ・ファイルを適用します．

カーネルの設定には，config，menuconfig，xconfigの3種類の方法があります．本稿ではmenuconfigを使用します（xconfigは使用できなかった）．

カーネルの設定手順を図3に示します．

● メモリとアドレス・マップの設定

メモリの設定とアドレス・マップの設定を行います．FPGAのプロジェクト内に生成されたPTFファイルを使用します．

コラム 開発環境について

FPGAボードには，Altera社の「Nios II開発キットStratixエディション」を使用しました．図Bにキットに付属するFPGAボードのブロック図を示します．今回使用するμClinuxディストリビューションでは，Nios II開発キットのCyclon，Cyclon II，Stratix IIの各エディションやサードパーティが提供しているFPGAボードもサポートされています．

開発用のパソコンとして，今回はWindows XP SP2とLinux（Fedora Core 6）の2台を用意しました．

Windowsパソコンには，Altera社のFPGA開発ツールQuartus II v7.1 Web EditionとMegaCore IPライブラリ，Nios II EDSを，Linuxパソコンには Linux版のQuartus II v7.1とMegaCore IPライブラリ，Nios II EDSをインストールしています．

Linux版のサポートOSはRed Hat Linux Enterprise 3 & 4となっていますが，本稿の範囲ではFedora Core 6で動作しています．ただし，ダウンロード・ケーブルについてはByteBlastor IIのドライバがインストールできなかったため，USB Blastorを使用しています．

LinuxパソコンとFPGAボードは，FPGAコンフィグレーション用のJTAGダウンロード・ケーブルで接続します．

図B 使用したFPGAボードのブロック図
Nios II開発キットStratixエディションに付属のFPGAボードを使用した．

図2
μClinuxを動かすまでの手順の概要
開発環境として，WindowsパソコンとLinuxパソコンが必要．

```
# cd uClinux-dist
# zcat /home/nios2_uClinux/uClinux-dist-20070130-nios2-02.diff.gz | patch -p0
# make menuconfig
```

ディレクトリの移動
Nios II 用パッチの適用
カーネルの設定

```
root@localhost:/home/share/nios2_uclinux/uClinux-dist
ファイル(E) 編集(E) 表示(V) 端末(T) タブ(B) ヘルプ(H)
uClinux v3.2.0 Configuration
Main Menu
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help.
Legend: [*] built-in [ ] excluded <M> module < > module capable
Vendor/Product Selection --->
Kernel/Library/Defaults Selection --->
---
Load an Alternate Configuration File
Save Configuration to an Alternate File
<Select> < Exit > < Help >
```

```
root@localhost:/home/share/nios2_uclinux/uClinux-dist
ファイル(E) 編集(E) 表示(V) 端末(T) タブ(B) ヘルプ(H)
uClinux v3.2.0 Configuration
Vendor/Product Selection
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help.
Legend: [*] built-in [ ] excluded <M> module < > module capable
--- Select the Vendor you wish to target
(Altera) Vendor
--- Select the Product you wish to target
(nios2nommu) Altera Products
<Select> < Exit > < Help >
```

① Vendorとして「Altera」，Productsとして「nios2nommu」を設定する．[Exit]で前の画面に戻る．

```
root@localhost:/home/share/nios2_uclinux/uClinux-dist
ファイル(E) 編集(E) 表示(V) 端末(T) タブ(B) ヘルプ(H)
uClinux v3.2.0 Configuration
Kernel/Library/Defaults Selection
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help.
Legend: [*] built-in [ ] excluded <M> module < > module capable
(linux-2.6.x) Kernel Version
(None) Libc Version
[*] Default all settings (lose changes)
[ ] Customize Kernel Settings (NEW)
[ ] Customize Vendor/User Settings (NEW)
[ ] Update Default Vendor Settings
<Select> < Exit > < Help >
```

② Kernel Versionで使用するカーネルのバージョンを選択する．今回は，バージョン2.6.19-uc1を使用するので，「linux-2.6.x」を選択する．Libc Versionで使用するライブラリを選択する．今回はインストールしたツールの中にあるものを使用するため，「None」を選択する．Nios IIのデフォルト設定を適用させるため「Default all Settings」を選ぶ(スペース・キーを使用)．すべて選択したら，[Exit]を押す．

```
root@localhost:/home/share/nios2_uclinux/uClinux-dist
ファイル(E) 編集(E) 表示(V) 端末(T) タブ(B) ヘルプ(H)
uClinux v3.2.0 Configuration
Do you wish to save your new kernel configuration?
< Yes > < No >
```

③ 設定を保存するために[Yes]を押す

図3　カーネルの設定手順

```
# make vendor_hwselect SYSPTF＝~/
standard/NiosII_stratix_1s10_standard.
ptf
```

「Please select which CPU you wish to build the kernel against」では，カーネルを実行するCPUを指定します．今回は「(1) cpu_class : altera_nios2 Type : S」を選択します．

「Please select a device to upload the kernel to」では，カーネルを保存するデバイスを指定します．今回は「(1) ext_flash class : altera_avalon_chi_flash」を選択します．

「Please select a device to execute kernel to」カーネルをロードして実行するデバイスを指定します．μ Clinuxを実行するには最低でも2Mバイト程度のRAMが必要になるので「(3) sdram class : altera_avalon_new_sdram_controller」を選択します．

アドレス・マップの設定のために，SOPC Builderから出力されたPTFファイルを指定します．これより，Make時にPTFファイルの情報からnios2_system.hが自動的に生成されます．

● カーネルのビルド

μClinuxカーネルを生成（ビルド）します．手順を**図4**に示します．作業ディレクトリのimagesの中にzImageファイルが生成されます．これがμ Clinuxシステムのイメージです．

今回の設定ではμClinuxカーネルが起動時に使用するルート・ファイルは，このイメージ・ファイルに組み込まれています[注3]．

● μClinuxシステムの起動

zImageファイルはSDRAMへダウンロードします．

FPGAボードの電源を投入すると，フラッシュ・メモリのコンフィグレーション・データによりFPGAに回路情報が書き込まれます．次にアプリケーション・データ（プログラム）がフラッシュ・メモリから読み込まれ動作を開始します（プログラムが書き込まれていない場合は動作しない）．この時点でダウンロード・コマンド（nios2-download）を実行するとSDRAMへダウンロードが開始します．

手順を**図5**に示します．sdk_shellを起動してダウンロードします．μ Clinuxカーネルが起動すると，ターミナルに起動画面が表示されます．

注3：ルート・ファイルはlinuxが起動するときに必要とするファイル・システム．主に必要なものとして/bin，/home，/etc，/devなどがある．これはmake実行時にμ Clinux-dist/romfs以下に生成される．またイメージ・ファイルへの組み込みについてはmake linux image実行時に行われる．イメージ・ファイルに組み込むか組み込まないかについては，カーネルの設定時に指定できる（デフォルトでは組み込む）．

```
# mkdir romfs      ← イメージ・ファイルを生成するためのディレクトリ作成

# make

# make linux image
```

図4　カーネルの生成

3. CFカードからμClinuxを起動する

今回使用したFPGAボードには，CompactFlash（CFカード）用のコネクタが実装されています．ここでは，μ ClinuxをCompactFlashから起動する方法を説明します．

● CompactFlashインターフェース回路の生成

これまで使用していたstandardプロジェクトには，CompactFlashの回路が含まれていないため，そのままで

```
# su
******                         ← パスワード

# sdk_shell

$ nios2-download -g images/zImage   ← ダウンロード

$ nios2-terminal                ← ターミナルを起動
```

```
uClinux/Nios II
Altera Nios II support (C) 2004 Microtronix Datacom Ltd.
Built 1 zonelists.  Total pages: 4064
Kernel command line:
PID hash table entries: 64 (order: 6, 256 bytes)
Dentry cache hash table entries: 2048 (order: 1, 8192 bytes)
Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
Memory available: 14080k/16384k RAM, 0k/0k ROM (1465k kernel code, 680k data)
Mount-cache hash table entries: 512
NET: Registered protocol family 16
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP est                        1024 (order: 0, 4096 bytes)
...
Command: mkdir /var/empty
Command: ifconfig lo 127.0.0.1
Command: route add -net 127.0.0.0 netmask 255.0.0.0 lo
Command: cat /etc/motd
Welcome to

For further information check:
http://www.uclinux.org/

Execution Finished, Exiting

Sash command shell (version 1.1.1)
/>
```

図5　μClinuxシステムのダウンロードと起動

図6　CompactFlashインターフェースの追加

Quartus Ⅱ のIPコア管理機能であるSOPC Builderを使う．CompactFlashインターフェース回路は標準で用意されている．ベース・アドレスはctlが0x800000，ideが0x800040，割り込みは7と8に設定する．

図7　CompactFlashの準備

使用するCompactFlashは，Linuxパソコンを使ってフォーマットし，データを書き込んでおく．

図9　CompactFlashにデバイス・ファイルをコピー

/>はμClinuxのプロンプト．

は利用できません．そこでCompactFlash用のハードウェアを追加します．

ハードウェアの生成にはWindows環境を使用します．standardプロジェクト・ファイルを作業フォルダにコピーし，プロジェクトを開きます．

CompactFlashインターフェース回路は，IPコアとしてNios Ⅱ開発システムに標準で付属しています．Quartus Ⅱ のIPコア管理機能のSOPC Builderを使い，「Compact Flash Interface（True IDE Mode）」を追加して，ベース・アドレスと割り込みを設定するだけです．今回はベース・アドレスはctlが0x800000，ideが0x800040，割り込みは7と8に設定しました（図6）．

CompactFlashインターフェースのピン配置は，FPGAボードの回路に合わせて指定します．ここでAssignments Editorを起動すると，Quartus Ⅱの以前のバージョンのものと思われる間違った信号名が追加されます．実際に使用する信号名はstandard_cfフォルダのNios Ⅱ_stratix_1s10_standard.bsfファイルに記述されているので，信号名を修正します．また，標準機能のLCDモジュールとCompactFlashのピンが重複しているところがあるので，LCDのピンをJ12からJ15へ変更します．

コンパイルを行い，完成したプロジェクト一式（standard_cfフォルダ）をLinux環境へコピーします．

● CompactFlashの準備

ルート・ファイルは，make実行時に生成されますが，/dev以下に必要なデバイス・ファイルだけは生成されません．デフォルト設定のときはμClinux起動時にromfs_listファイルから生成しています．CompactFlashから起動する場合は，これらのすべてのファイルをCompactFlashに保存する必要があります．そこで，CompactFlashのドライバを追加したμClinuxカーネルを生成・起動し，CompactFlashをマウントして/dev以下のファイルをコピーする必要があります．

LinuxパソコンにCompactFlashを接続します．筆者の環境では/dev/sdaと認識されました．オート・マウントされた場合はアンマウントしてください．手順を図7に示します．これでルート・ファイルはほぼでき上がりです．

次に/dev以下のデバイス・ファイルを用意します．まず，FPGA単体で動作するμClinuxにCompactFlashのドライバを組み込みます．手順を図8に示します．また，FPGAの回路を変更するため，フラッシュ・メモリを書き換えるか，SOFファイルをFPGAに直接書き込みます．zImageファイルをダウンロードしてμClinuxを起動します．図9の手順で，/dev以下のデバイス・ファイルをCompactFlashの/devにコピーすれば，CompactFlashの準備は完了です．

● CompactFlash起動のカーネルのビルド

CompactFlashから起動するカーネルを作成します．図8と同様に，

```
# make menuconfig
```

で設定を行います．Kernel/Library/Defaults Selectionで
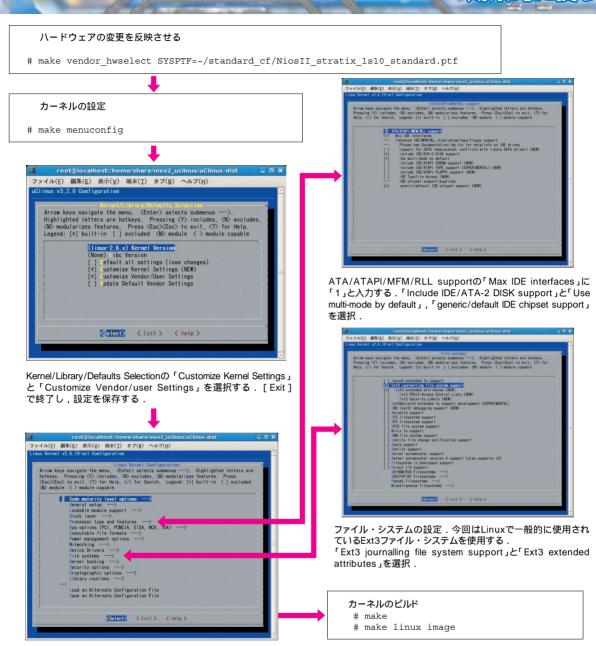
図8　CompactFlash ドライバ付きのカーネルのビルド

は「Customize Kernel Settings」にチェックを入れ，［Exit］で設定を保存します．カーネルの設定メニューでは，Processor type and featuresのDefault kernel command stringに「root=/dev/hda1 rw」と入力します〔**図10(a)**〕．また，ramfsの設定を解除するためGeneral setupの「initramfs source file(s)」の記述を削除します〔**図10(b)**〕．

● CompactFlashからμClinuxを起動

zImageファイルをダウンロードして，コンソールに**図5**と同様のメッセージが表示されれば起動完了です．

動作の様子を**写真1**に示します．LCDに文字が表示されていますが，これは筆者が別のボード用(汎用マイコン)にとりあえず動かすために作ったLCDドライバを移植したものです．カーネルの設定メニューに「Nios LCD 16207 device support」という項目があります．makefileを読んでみたところ，チェックを入れてもドライバは組み込まれないようなので，自作のものを組み込みました．

● まとめ

今回はすでに用意されているIPコアやデバイス・ドライ

(a) Processor type and features

(b) General setup

図10 CompactFlash起動のカーネルの設定

図8と同様にビルドする．図8④の設定メニューからの設定内容を示す．
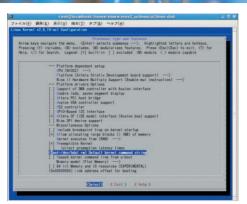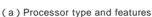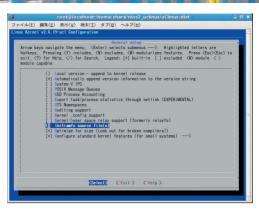
写真1 μClinuxが動作しているFPGAボード

バなどを使用したため，比較的簡単に実装できました．実際に組み込み用途で使用する場合には，さまざまな周辺機能が接続されます．現在はNios IIのパラレルI/Oを使用してリアルタイム・クロックの接続を行っていますが，今後はUSBなども試したいと思っています．

参考として，今回実装したバイナリ・ファイルの大きさを**表1**に示します．

本来のLinuxであれば，ブート・ローダ（GRUB，U-BOOT，Redbootなど）を使用して，Linuxのバイナリ・ファイルをハード・ディスクなどからロードし，RAMへ展開して起動します．μClinuxでも同様ですが，今回は直接SDRAMへバイナリ・ファイルをダウンロードして起動しています．

表1 バイナリ・ファイルの大きさ

カーネルの設定	バイナリ・ファイル（Kバイト）	備　考
デフォルト設定	1,213	ルートを含むため容量大
CFルート	798	

また多くの場合，Nios IIのアプリケーション・ソフトウェアは，外部フラッシュ・メモリからFPGA内部メモリまたは外部メモリ（SDRAMなど）に展開して動作します．このため，FPGAの起動時にμClinuxも起動します．筆者もμClinuxのバイナリ・ファイルをフラッシュ・メモリへ書き込んで動作確認しています(6)．

Linuxを使えば，ネットワークを活用するシステムの設計効率が上がります．本システムをさらに活用したシステムについては，機会を改めて紹介したいと考えています．

参考・引用*文献

(1) Nios Development Board Reference Manual, Stratix Edition, ver 1.1, Oct. 2004.
(2) Quartus II Handbook Volume 5: Embedded Peripherals, ver 7.1, May 2007.
http://www.altera.co.jp/literature/lit-nios2.jsp
(3) uClinux-dist Developers Guide Version 1.3.0.
http://suzaku.atmark-techno.com/downloads/all/&dir=/doc
(4) Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman, LINUXデバイスドライバ，第3版，オライリージャパン．
(5) Nios Community Forum, http://www.niosforum. com/index.php
(6) Nios Community Wiki, http://nioswiki.jot. com/WikiHome

なかね・たかやす
(株)ネクスト・ディメンション
ふくしま・まさふみ
東京計器工業(株)